Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 10 (2012) 414 – 421

Procedia Computer Science

The 3rd International Conference on Ambient Systems, Networks and Technologies (ANT)

Leveraging Seminal Protocol Stacks to Support MANETs

Mohamed Amine ABID[a], Abdelfettah BELGHITH[b]

HANA Research Group

[a]*mohamed-amine.abid@hanalab.org*
[b]*abdelfattah.belghith@ensi.rnu.tn*

Abstract

In this paper, we investigate the suitability of some protocol stacks (mainly the UDP/IP model) that were firstly proposed for wired networks, to the mobile ad hoc networks (MANETs). These latter, are known to be dynamic as the topology is constantly changing, which affects the validity of the established routes as time tics farther from the start of every routing period. In the UDP/IP model for instance, the IP layer forecasts an IP packet as soon as it receives a data unit coming from the upper layer independently of when this packet will effectively be sent by the lower layer (i.e., the underlying MAC). As such, if a newly created packet using a routing decision relative to its instant of creation, is delayed at the lower layer for some reason, it will be sent using an out of date routing decision and then will wander through established invalid routes leading to poor performances. We first investigate the inherent functioning of such a model (firstly proposed for static networks) and exhibit its drawbacks in a MANETs context. Then, we propose practical enhancements to accommodate a proper behavior suitable to dynamic networks and that firstly provides a priority treatment to control traffic and secondly makes the routing decisions (fills up the IP next hop field) only when the MAC is ready to handle the packet. We conducted an extensive set of simulations to compare both implementations using different load and mobility scenarios and performance metrics but the same routing protocol (OLSR). These simulations show that our proposal solves the malfunctioning of current implementations, yet it allows a better behavior of the OLSR protocol and thus more accurate network performances.

Keywords:
MANETs, Routability, Mobility, Load

1. Introduction

When we first start interest on networking, we were rather concerned in interconnecting devices using wires. Different technologies were used, but all consist on physically connecting distant peers. Research communities proposed a variety of architectures and protocols aiming to allow distant computers for instance, to communicate through wires. As the interest on the networking field increased, new concepts were presented substituting older ones, such it was the case of moving from the circuit switching to the packet

[1]Corresponding author. Tel.: +216-20-925-974. E-mail address: mohamed-amine.abid@hanalab.org

1877-0509 © 2012 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.procs.2012.06.054

switching firstly proposed by Leonard Kleinrock on July 1961 [1]. All these proposals have finally led to the definition of the standard protocol stack (the reference model) or some real used stacks such the UDP/IP model which is by the way, the most deployed model for wired communication between distant nodes. Every layer in the protocol stack addresses a known and well defined problem facing this wired communication between distant computers. The main characteristic of these wired networks is certainly their static aspect. The network topology is maintained the same during time. A change in the topology is rather due to a problem occurrence (a node getting down for example).

The technological development brought networking into a new era: the emergence of wireless networks. The main advantage of such type of networks is the possibility for a node to be mobile. We can distinguish two types of mobility: mobility of the last hop node, where only the end user could be mobile, while the rest of the network (the core) is rather fixed and wired. In this case we are speaking about infrastructure wireless networks such as the cellular networks. The other type is mobile ad hoc networks, commonly called MANETs, where every node is mobile. No infrastructure is deployed and nodes are intended to cooperate to route exchanged information between every pair of source-destination nodes in the network. MANETs are rather dynamic. The topology changes are no longer a consequence of an occurring problem solely as it was the case with the wired static nets. Routing protocols are then used to track the topology changes over time, and hence populate routing tables with correct routes from a source towards every destination node. Of course, since the evolution of the topology of a mobile ad hoc network is an inherent behavior, the frequency of routing table computation is much higher than the wired case (we are speaking here about proactive routing protocols). This frequency is normally closely related to the mobility degree of the network, but it is usually fixed by the underlying used routing protocol. If this protocol does not adapt the updating period size to the mobility level, the same fixed period size will be always used.

One of the most known routing protocols for MANETs is the Optimized Link State Routing Protocol (OLSR[2][3][4]). OLSR is a proactive link state protocol. In LS routing, Link-state advertisements need to be spread across the entire network while routing table advertisements in distant vector (DV) are only for immediate neighbors. Compared to link-state protocols, distance-vector routing has less computational complexity and signaling overhead [5]. The OLSR routing protocol uses Hello and Topology Control (TC) messages to collect link state information and build the topology of the network. Hello messages are used to discover the 2-hops neighborhood and elect a set of multipoint relays (MPRs). MPRs elected by a node, are used to advertise that node outside its neighborhood. OLSR owes its name to that optimization. Its aim is to reduce the signaling overhead by reducing redundant information. The problem is that each node uses its own view of the topology built from the received control messages. A delayed control message leads to an out of date topology construction, and thus to a wrong routing decision.

The transition from the wired case to the wireless case was rather made by keeping the protocol stack almost unchanged. The only modified layers were lower layers, namely, the physical/link layers. The rest of the stack was kept the same as firstly implemented for the wired networks. As aforementioned, most of the real implementations use the UDP/IP model. A mobile ad hoc node is considered through this point of view, as any common host/router in the wired case. Applications are run above TCP or UDP (transport protocol). The used network and link layers are IP and MAC respectively. All these protocols interact with their neighboring layers by the same old way. The question that obviously surfaces is then the following: is it enough to change the Link/physical layers and just keep upper layers the same? Does the wired protocol stack suite the dynamic wireless networks?

In this paper we are investigating this adequacy by focusing on the particular interaction between the IP and MAC layers. We will take the example of the implementation used in the INET framework of the OMNeT++ [6][7][8] simulator as an illustrative case. We will show through simulation that using this old way of communication, inherited from the wired protocol stack, is problematic for the mobile ad hoc case. A new implementation is then used and compared to the old one, through the deployment of the same network scenario using the OLSR routing protocol.

2. Influence of mobility and load on The IP/MAC Communication

2.1. disadvantages of using the wired implementation in a mobile ad hoc canvas

In the traditional UDP/IP protocol stack, henceforth called the current Model, a message generated by the application (upper layers) is immediately passed to the network layer (to IP). A new data packet is then prepared encapsulating this message (or a fragment of it). IP fills out the different fields of the packet by the adequate information, selects the Next Hop address to be used (the gateway) using the current routing table of the node, then puts the resulting packet at the end of the transmission queue (called *the management queue* in OMNeT++). While MAC has not yet claimed a new packet (it is already dealing with an old packet), *Management* keeps all packets coming from the IP, and serves them one after the other (FIFO), at each MAC's request (MAC being the server of our queue).

Such an implementation may rather suit wired networks; or more generally static networks without mobility. In such networks, the established routes remain almost the same throughout time. So, even if the routing decision is made long before the instant of transmission, it may remain correct since the topology is static. However, in Ad hoc mobile networks, routes may change continuously and deeply from one period to its following. Even at the end of a routing period, the network topology may be deeply different from its start. The collateral side effects of this particular behavior could be nasty to the functioning of our network. In fact, when a node keeps using the same routing decision made when the packet was firstly created, during the entire period, it may try to send it through a broken link. This results on a waste of our network resources because of the performed retransmissions. Furthermore, the packets in the management queue will be blocked until the currently treated packet at the MAC layer is served (whether correctly sent or discarded after reaching the limit of retransmissions). Thus, blocked packets created at a given period i (using routing decision of period i), could be sent during periods j, where $j >= i$.

Every data packet transmitted through a broken link, will simply block all the data packets awaiting at the *management* queue. This waiting delay may be aggravated by the exponential back-off mechanism used at the MAC layer. And as a consequence, the whole problem will be simply exported to the packets in the transmission queue. We will be in a situation where the *management* queue becomes soon full of packets (whether locally generated or to be forwarded) ready for sending, but which were prepared or created in the past using the routing table of the moment of their arrival at the IP layer, which may probably represent a different topology from the moment where they will be actually sent.

Worse yet, even if we receive the new wave of announcements that corrects the routing tables, these packets, already in the management queues, will not be affected: the routing decisions are already made for these packets. If all packets are treated alike (Hellos and data packets), even advertisements will be sent in late; they will announce already outdated routes and consequently make the situation even worse.

2.2. The proposed correction

Our problem can be summarized in when to make the routing decision? (i.e. when should we select the *Next Hop* address to be used when treating a packet to be sent?). This particular decision should be made once the packet is claimed by the MAC layer for its immediate transmission. Hence the need to change the position of the waiting queue, currently below IP, to an intermediate layer above IP. When the MAC requires a new packet for immediate sending, the IP module takes the head of the transmission queue (now over the IP module), creates the packet, and selects in particular the *Next Hop* using the current routing table (the latest information we have), and finally upload it to MAC. The routing decision should only be made when the packet is considered for immediate sending.

Such a proposal may quickly overcome the situation of blocked packets in the transmission queue because of a packet transmitted through a broken link, but certainly do not avoid it. In fact, when the transmission queue is removed from its old position relatively to IP, to the new position over IP, all what is done here is that we ensure the use of our latest available routing information before sending a data packet. But, this latest information detained by a node may not be the freshest one. The spread of a new routing information throughout the entire network may be slowed down by data packets at the transmission queues. Since we are treating the control traffic and the data one alike, the data load applied to the network will have a great

influence on the updating process and therefore on the network performances. Routing messages could always be blocked by awaiting data packets, and may then reach nodes in the network in late. Entries in the routing tables will be out of date, making the blocking period gradually greater. Changing the transmission queue position will simply avoid the situation where a packet is sent using a routing decision made in the past. It only guarantees that every node uses the latest routes it has, to send data packets and certainly does in no way mean that these routes are the freshest or valid.

To enable at all costs, the establishment of the maximum of valid routes at the start of each routing period, we have to accelerate the updating process by eliminating the old aforementioned substantial delays caused by the awaiting data packets. As such, we propose to give Hello messages the priority over normal data traffic. Hello messages have to be transmitted as soon as possible before any other awaiting data packet. A simple modification of the IP algorithm makes it work with priority. Received and locally generated Hellos are then put at the head of the transmission queue in front of the awaiting data packets (the queue is no longer FIFO). Our objective here is rather to circulate the routing information as soon as possible and surely not to ensure any persistence of the routing validity as time progresses. This certainly depends on the level of the network dynamics and is currently under investigation. This new context of priority IP handling makes data traffic almost transparent to the updating process. A hello message needs only to wait for the current message being transmitted if any.

Notice here that using a priority queue with the old implementation doesn't solve the problem. The the prioritization mechanism only makes updating waves came in time. But already prepared data packets using the old routing information and that are awaiting in the transmission queues will not be affected. They will be routed using out of date routing information dating back to their creation moment. Even new created packets (using the newest routing information) are not sent immediately; they still need to wait for the consumption of all the old data packets in the front of the transmission queue, which probably needs 7 (by default) successive failed retransmissions for each packet, accompanied by an exponential backoff mechanism, to be eliminated.

Moreover, it is important to notify that this problem can't be avoided by simply changing the buffer size of the transmission queue. By reducing it, the transmission queues will be quickly saturated by data packets, and no place will be left for the signaling traffic to update the routing tables. In the other hand, the increase of the buffer size (including the extreme case of infinite size), will simply be reflected by the presence at the transmission queue of updating waves separated by data packets for which the routing decisions were already made. After discarding all the data packets in front of the first updating wave (which may take a long time because of the exponential backoff and the contention for the channel, making the topological information of this wave obsolete), this latter will be spread through out the network changing the routing tables by establishing out of date paths. The impact of such a behavior may be greater in this particular case. In addition, all this discussed problem is independent of the used routing protocol or the data packet size. It only concerns the protocol stack architecture, and especially the IP/MAC interaction, which is not suitable for dynamic networks.

3. Comparison between the Two Implementations

To compare the two implementations, we used simulation performed on the OMNeT++ simulator. We considered the following two metrics: the average number of the received packets per host, representing the ensured throughput by the network; and the average end-to-end delay (in sec) per received packet. Both metrics are plotted as a function of the applied data load, and for different node speeds. Other metrics should be considered, but here, we will limit ourselves to these because of lack of space.

3.0.1. Simulation Set Up

To establish this comparison, we conducted an extensive set of simulations. We have considered a simulation area of 1000m by 1000m, with 100 mobile nodes using the Random Way Point mobility model [9]. The simulation surface was divided into 9 areas, each of which contains from 11 to 12 nodes. The Random Waypoint mobility is then applied separately inside each area (Fig.1). This is basically to mitigate the known middle concentration effect of the Random WayPoint mobility model.

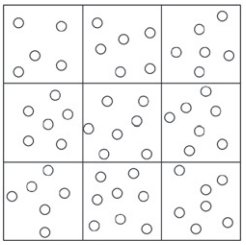

Fig. 1: Restricted Random Waypoint: Applied inside each area

The Transmission Range and the Carrier Sense Range are both set to $250m$. In all scenarios, we used a network capacity fixed to 54 Mbps and a maximum retransmission count equal to 4. Recall here that when we use our proposal, the only case where a data packet can block a control message is, when the currently treated data packet at MAC, is sent through a broken link. In this particular case, control packets will be delayed by a maximum of 4 failed retransmissions, performed with the exponential back-off and aggravated by the contention time. In a deeper study, but not presented here, we showed that the value of this particular parameter is of a very high importance. We showed that using the common value of 7 retransmissions, crashes our network performances, even when we use our proposal. A very acceptable behavior is shown when we use the value of 3 maximum retransmissions. As a proactive routing protocol, we used the Optimized Link State Routing Protocol (OLSR[2][3][4]). The simulation transient regime is evaluated to 100 seconds, and the routing update periods of Hello and TC are equal to 2 and 8 seconds respectively. We ran our simulation during 300 seconds. We immobilized 10 source nodes at the left edge of the simulation area (randomly distributed over the three sub-areas at the left edge) and similarly 10 destination nodes at the right edge. We also considered 10 traffic flows using these fixed source-destination pairs (pairs are chosen randomly). As such, we insured multi-hop routes (4 to 5 hops) between source and destination pairs. We fixed the data packet size to 200 bytes. Finally the Max duration period, representing the life time of an entry in the routing table, is set to 30 seconds.

3.0.2. Simulation Results

As said before, we will base our comparison on two metrics: the throughput and the end to end delay. Figures (2.a) to (2.e) represent the average number of received packets per destination node, plotted as a function of the applied data load, and for the speeds of $1m/s$, $2m/s$, $5m/s$, $10m/s$ and $20m/s$ respectively. We can observe that using OLSR in the old implementation (the wired UDP/IP model) yields to lower performances than when it is deployed in our new proposed implementation. In our implementation, this same routing protocol (OLSR), ensures an average throughput almost equal to twice the performance when applied in the old architecture. In figure (2.b) for instance, at a speed of $2m/s$ and an applied data load of $30packets/sec$, the average throughput of the network using our implementation is 250% greater than the one using the current implementation. It is clear that as the mobility level increases, the network performance gets worse. Recall here that we are not proposing a new routing protocol that leads to a better routability and then to better performances; but, we are rather using the same routing protocol in two different implementation canvas (the wired implementation vs. a new implementation that take into account the mobility aspect of MANETs). When the mobility level is high, the routability (i.e. percentage of valid routes) of OLSR will certainly be worse: it quickly and deeply relapses from its highest level reached at the very beginning of a routing period. This would explain the decrease of the average throughput when the speed increases (even for our implementation). A better routing protocol would yield to better results. For instance, if we use a routing protocol that selects stable routes rather than shortest ones (like in [10] [11]), or a protocol that autonomicly calibrates its routing period size according to the network dynamics (like the protocol in [12][13] or the one proposed in [14][15]) in a way to keep the routability at a high level during the entire routing period (the routability at the end of the routing period is kept the same as at its start); and since we only use the latest and freshest information we have, all the data packets will be routed through valid routes leading certainly to better performances.

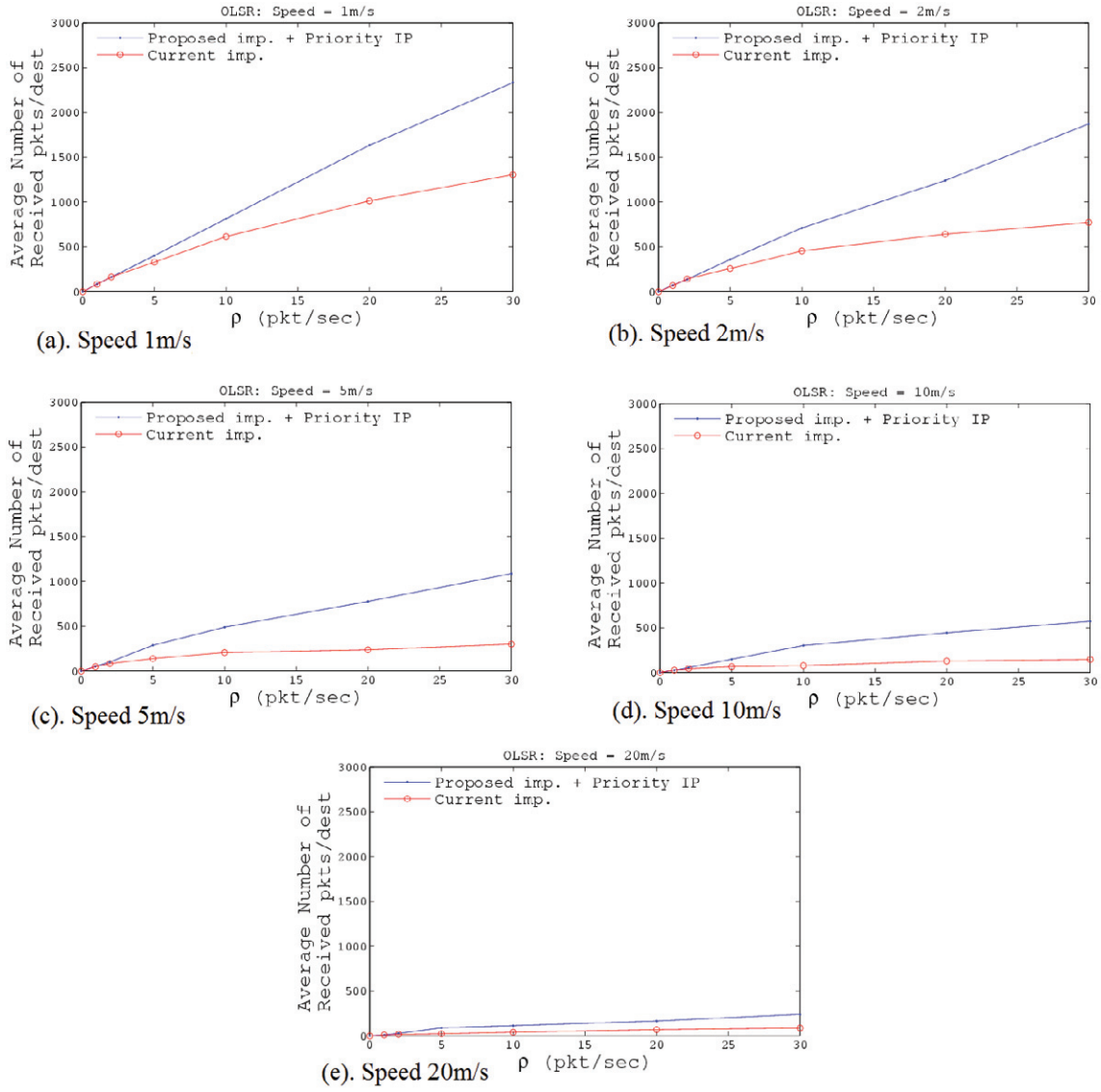

Fig. 2: Average number of received packets/dest

For both implementations, when nodes move slowly, a slight delay formed when Hellos are received, leads to an acceptable routing. Selected routes may than remain correct for mostly the entire routing period. Data packets travel rapidly through these routes until reaching their ultimate destinations. The average end-to-end delay is consequently small and acceptable and do not exceed some few milliseconds. For the new implementation, this end-to-end delay start increasing, even at low speeds, when the data load applied to the network gets higher (an end-to-end delay of almost $1sec$ at a speed of $1m/s$ and a data load of $30packets/sec$ (Figure 3)). Such a behavior is rather expected since our topology changes slowly over time (low speed), and consequently, only packets sent at the end of a routing period could experience failure. When the applied data load gets higher, the fraction of packets routed through outdated links gets higher too, leading to a slight increase of the end to end delay. As the speed gets higher, the routability falls deeper and faster over time. Depending on the applied speed, the topology of our network changes more or less deeply compared to the start of the current routing period. The aforementioned description of how the current implementation causes

the blocking of data streams, and thus the Hello messages too, should be easily detected in high or moderate scenarios (where the routability is rather low early before the end of a routing period). The resulting end to end delay of the current implementation is then much higher than our proposed implementation. At a speed of $5m/s$ and an applied data rate of $30packets/sec$, our implementation yields to an end to end delay of $2seconds$, while it reached $28seconds$ with the current implementation. Notice here that even when we used our implementation, a significant end to end delay is detected. This delay is also caused by a blocking data packet, which is not at the transmission queue (now with priority), but rather at the MAC layer. As said before, the only data packet that could be sent before a control message, is the one already treated at the MAC layer. To overcome such an undesirable behavior, we can use a preemptive MAC layer to avoid this situation, or more simply reduce this blocking period through the use of a smaller limit number of retransmissions (We get a better results with a maximum retransmission count equal to 3).

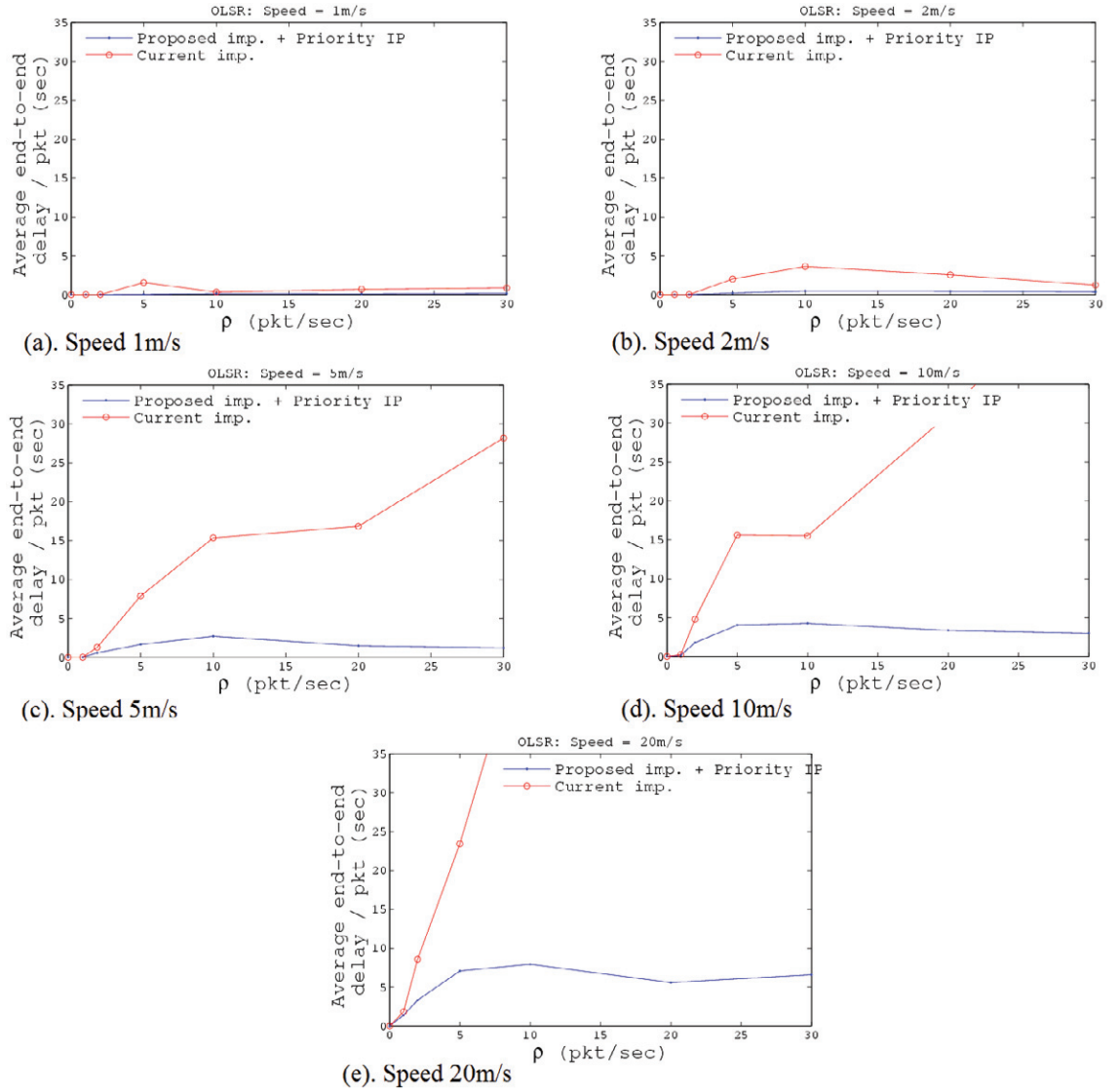

Fig. 3: Average number of received packets/dest

4. Conclusion

In this paper, we show that the current architecture of the Internet (IP/UDP) protocol stack, firstly made for wired networks, is not adequate for the ad hoc mobile networks. When the network is dynamic, mobile nodes are unable to track the topology changes in real time, which depletes the network for nothing. We then proposed a new architecture with the ability to quickly overcome this vulnerability, without preventing its occurrence. This proposal was then augmented by giving a priority to Hello messages. As such, the control traffic will no longer be blocked by the data load ; and consequently, the routability will increase to its upper limits, and avoids the useless retransmissions wasting the network resources.

Our proposal was then compared to the already used current model through simulation. We showed that for the same tested routing protocol (OLSR), the network yields to better performances when deployed using our implementation than when we use the old one.

References

[1] L. Kleinrock, Information flow in large communication nets, Rle quarterly progress report, Massachusetts Institute of Technology (July 1961).
[2] T. Clausen, P. Jacquet, Optimized link state routing protocol (olsr) (2003).
URL `http://citeseer.ist.psu.edu/clausen03optimized.html`
[3] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, , L. Viennot, Optimized link state routing protocol for ad hoc networks, Proceedings of the IEEE International Multitopic Conference (INMIC 2001) (2001) 1374 – 1378.
[4] T. Clausen, G. Hansen, L. Christensen, G. Behrmann, The optimized link state routing protocol, evaluation through experiments and simulation, In Proceedings of the IEEE Symposium on Wireless Personal Mobile Communications.
[5] G. E. R. D. C. Vasiliadis, C. Vassilakis, Performance evaluation of distance vector routing protocol on a wireless circular model, Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics, Springer Netherlands (2008) 323–328.
[6] A. Varga, Omnet++: Discrete event simulation system version 3.3, user manual, Tech. rep. (Last updated: March 29, 2006).
[7] G. Pongor, Omnet: Objective modular network testbed, in: MASCOTS '93: Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems, The Society for Computer Simulation, International, San Diego, CA, USA, 1993, pp. 323–326.
[8] A. Varga, The omnet++ discrete event simulation system, Proceedings of the European Simulation Multiconference (ESM'2001).
[9] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002.
[10] A. B. M. Abdelfettah Belghith, M. A. Abid, Suitability analysis of probabilistic routing for dynamic ad-hoc networksThe 5th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (IEEE WiMob 2009), Marrakech, Morocco.
[11] M. A. A. Abdelfettah Belghith, A. B. Mnaouer, Adaptive probabilistic proactive routing for dense manets, Journal of Computer Networks and CommunicationsHttp://www.hindawi.com/journals/jcnc/aip/234824/.
[12] A. Belghith, M. A. Abid, Autonomic self tunable proactive routing in mobile ad hoc networksThe 5th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (IEEE WiMob 2009), Marrakech, Morocco.
[13] A. Belghith, M. A. Abid, Dynamically self adjustable proactive routing protocols for mobile ad hoc networksThe 34th Conference on Local Computer networks LCN 2009, Zurich, Switzerland.
[14] M. A. Abid, A. Belghith, Period size self tuning to enhance routing in manets, International Journal of Business Data Communications and Networking (IJBDCN) 6 (4) (2010) (21–37), special Issue on Future Trends on Ad hoc and Sensor Networks (FT-ASN).
[15] M. A. Abid, A. Belghith, Asynchronous locally self adjusted routing protocol for mobile multi hop ad hoc networksThe 8th ACS/IEEE International Conference ACS/IEEE AICCSA'10, Hammamet, Tunisia.